Modeling and Capturing Users' Actions in CSCL Systems for Collaboration Analysis Purposes

doi:10.3991/ijet.v4s1.736

R. Duque[1], C. Bravo[1], J. Gallardo[1], W.J. Giraldo[2] and M. Ortega[1]
[1] University of Castilla – La Mancha, Ciudad Real, Spain
[2] University of Quindío, Quindío, Colombia

***Abstract*—A significant number of CSCL (Computer-Supported Collaborative Learning) environments support the learning of groups of students enabling their collaboration in solving problems. These collaborative environments usually need additional computational support to allow the automatic processing of both the actions carried out by the students and the end solution with the aim of studying the learning process and the validity of the solution proposed to the problem. This process, known as Collaboration and Interaction Analysis, is typically carried out in three phases: observation, abstraction and intervention. In this paper, we propose a methodological approach for the design of mechanisms for the observation phase. This approach provides a set of procedures enabling developers to design observation systems in CSCL environments that capture and model all the information required for comprehensive analyses of the collaboration process and the resulting solution to the problem. This methodological approach is put into practice by means of its use in the design of an observation system in the SPACE-DESIGN (SPecification and Automatic Construction of collaborative Environments of DESIGN) collaborative environment.**

***Index Terms*—CSCL, Collaboration and interaction analysis, Observation mechanisms.**

I. INTRODUCTION

CSCL (Computer-Supported Collaborative Learning) environments provide shared workspaces where students carry out learning activities using tools for learning object manipulation, discussion and coordination. A significant number of such environments are aimed at collaborative problem solving [9], where the teacher proposes a problem to be solved by the students through collaborative work within a group. In this context, there is a clear need for a computer-supported analysis process to characterize the type of collaboration undertaken by the students and its influence in the learning process [8] as well as the benefits provided by collaboration in the learning process [15].

These collaboration and interaction analysis processes start off with an observation phase where the actions carried out by the users of the CSCL environment and the solution under construction (typically a document or artifact) are modeled and stored. This is followed by an abstraction phase, in which a set of analysis indicators (variables) [3] are inferred from raw data from the previous phase. These indicators describe aspects such as the quality of the students' work, the properties of the solution proposed to the problem and the way in which a student (or a group) uses the CSCL environment to communicate/coordinate with others. In a final phase of intervention, these indicators are used in different forms to improve the learning process (e.g., to send advice, to automate evaluations, to change the system's behavior, etc.).

Many CSCL environments incorporate automatic analysis mechanisms [18]. However, most of them approach only the analysis of collaboration and do not deal with the analysis of the model or artifact built by the students. This fact is probably due to the lack of methodological guidelines for the design of observation mechanisms to model and capture the collaboration process together with the solution to a problem. This paper presents a methodological approach that fills the aforementioned gap with a set of procedures and techniques for the design of observation mechanisms to process (capture, represent and store) all kinds of significant information coming from the use of a CSCL environment following a problem-solving approach to learning. This is oriented to software developers, so that they have a technological support to facilitate their work of building interaction automatic observation systems. The proposed methodological approach is based firstly on the specification of models to represent the elements involved in collaborative problem-solving tasks and secondly on the process-solution analysis framework proposed by Bravo et al. [2].

The article unfolds in four additional sections. Section II analyses the main research works in the field of design of automatic observation mechanisms in CSCL environments. Section III describes our methodological approach for the design of observation mechanisms for analysis purposes. Section IV presents a case study where the approach is used to design an observation system for the SPACE-DESIGN (SPecification and Automatic Construction of collaborative Environments of DESIGN) collaborative environment. Section V discusses the conclusions drawn from this work and proposes future lines of work.

II. RELATED WORK

Most approaches to the automatic analysis of students' work within CSCL environments have focused on processing the repositories of raw data representing the communication and coordination interactions of the users [17]. Li et al. complements the use of these sets of raw data with ontologies [10] to conceptualize the elements involved in the problem solving process and, in so doing, enable an analysis of this work process to be made. However, a more complete analysis requires the

processing not only of sources of information about communication and coordination but also of interactions aimed at building the solution. This issue is approached by Martinez et al. [12], who define the structure of repositories of actions using XML-based languages to store the interactions of the users. In the same way, Avouris et al. [1] define five observation dimensions to model the actions stored in these repositories. These dimensions are the following:

- Temporary: Timestamp of the interactions.
- Actors: Users who are involved in solving the problem.
- Attributes: Elements used by the users to solve the problem.
- Events: Activities of the users.
- Comments: Additional information to complete the capture of actions.

A second set of works approaches the design of analysis tools to process the repositories of actions to characterize the collaborative work [6]. According to the principles guiding the creation of analysis tools proposed by Miller et al. [14], these analysis tools should generate both relational and content information about the collaborative work of the students. This is the case of *ActiveMath*, a web-based learning environment that uses a set of predefined methods to generate automatically an evaluation of the interaction between students [16]. However, these proposals do not deepen into methodological guidelines for developing observation systems that capture the actions carried out by users and classify them for analysis purposes according to computational models.

The framework proposed by Bravo et al. [2] aims to deal with some of the aforementioned limitations. It proposes a taxonomy of actions that the student can carry out (oriented to problem solving, coordination, etc.). In addition, this framework provides a conceptualization for analysis that considers both the collaborative work and the solutions built, separately and jointly. This paper takes this framework as a basis and complements it by means of a methodological approach that relies on the building of computational models and procedures for the design of observation mechanisms in CSCL environments, and furthermore that ensures criteria of completeness (processing both the collaborative work and the solution made) and of integrity (processing both the group work and the individual activity).

III. METHODOLOGICAL APPROACH FOR DESIGNING OBSERVATION MECHANISMS

The methodological approach for designing observation mechanisms includes four procedures as shown in Fig. 1. The first one aims at identifying the actions that the students can carry out using the user interface. The second procedure proposes a number of steps for identifying the tasks that the students can carry out to solve the problem and, as a result, for designing the structure of the repository aimed at storing the users' activity expressed as a set of actions organized into a set of tasks. The third procedure enables an abstract solution model to be specified. The last procedure specifies the steps to be taken in order to implement the observation mechanisms.

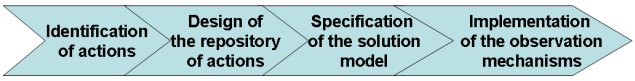

Figure 1. Procedures of the methodological approach.

A. *Identification of actions*

In this procedure, the developer of the observation system is helped by experts (teachers and evaluators) in order to start the design of a *diagram of actions.* The *diagram of actions* is a hierarchical model made up of five levels. The first four levels are defined in this procedure and the fifth level is completed in the next procedure of the methodological approach (see sub-section III.B).

The developer identifies the users' activity throughout the interaction with the CSCL environment and expresses it as a set of collaborative actions. An action is the most basic unit of analysis. According to Martinez et al. [13], a collaborative action is defined as "*an action that affects or can affect the collaborative process […]. The action itself or its effect can be perceived by at least a member of the group distinct of the one that performed the action*". The *diagram of actions* represents the collaborative and individual actions and classifies them.

The *diagram of actions* contains: in the top level, a root element that represents the CSCL environment; in the second level, the collaborative tools integrated into the environment; and, in the third level, the actions that each collaborative tool supports according to the following taxonomy:

- Instrumental: An action that inserts, modifies or deletes an instance (significant information element) in the shared workspace where the solution is created.
- Cognitive: An action that allows the extraction or management of some type of knowledge from the work carried out.
- Communicative: An action oriented to the exchange of ideas between the members of the group.
- Formal: An action that supports the development of the collaborative process in an ordered, synchronized way according to the collaboration policies established for the task.

The aforementioned taxonomy extends the criteria of Bravo et al. [2], which differentiates between actions that change the solution (problem solving actions) and actions that enable collaboration between users but do not interact with the solution (communication, coordination and decision-making actions). On the one hand, our taxonomy distinguishes between actions that interact with the solution to change its state (instrumental) and actions that interact with the solution to extract any kind of knowledge (cognitive). On the other hand, the collaboration support actions are classified as those that focus on discussion (communicative) between users (which traditionally have been the focus of attention of all the analytical work in collaboration analysis [11]), and those that enable collaboration with other mechanisms (formal), such as a proposal of task allocation or a request for floor control.

Finally, regarding the fourth level of the hierarchy, the developer assigns a name to each specific action within a specific type. Table I shows the levels and nodes used in the notation of the *diagram of actions.*

TABLE I.
FOUR LEVELS OF THE DIAGRAM OF ACTIONS

Icon	Level	Node
	1	CSCL environment
	2	Collaborative tool
	3	Instrumental action
	3	Cognitive action
	3	Communicative action
	3	Formal action
	4	Action identifier

In order to provide support to observation system developers, a CASE tool has been implemented. This CASE tool supports the design of *diagram of actions*; this automatically stores a representation of the diagram designed in the XMI[1] language (see Fig. 3). This CASE tool has been implemented using the GMF[2] environment that allows the development of authoring tools, integrating them into the Eclipse[3] development environment.

B. *Design of the repository*

This procedure takes the *diagram of actions* designed in the previous procedure as input. This procedure relates the actions with the tasks to be carried out by the students. As a result, the procedure provides the structure of a repository that stores all the information about the development of the users' activity in the form of actions and tasks. This goal is achieved by means of three steps.

In the first step, the developer identifies the tasks that should be carried out by the students. This consists of dividing the problem-solving process into sub-processes. These sub-processes are smaller or lower level processes, which produce meaningful information (e.g., an artifact or a document) and that is usually an input for a subsequent task. For example, a problem-solving process for Programming learning could be divided into three tasks: source code editing, compilation and execution. While the tasks produce results of interest for the users and for the systems, the actions are the result of the interaction of the student with the user interface of the CSCL environment (e.g., to create an object in a shared editor, to modify the properties of an object, etc.).

In the second step, the actions are linked with the tasks. A task is developed through a set of actions. In this step, the developer identifies which actions from the *diagram of actions* are characteristic of each task (e.g., to carry out a source code editing task will require instrumental actions in a shared editor).

The developer uses the aforementioned CASE tool (see Fig. 3) to draw the tasks (see Table II) identified into the *diagram of actions* (fifth level of the hierarchical model).

[1] www.omg.org/technology/documents/formal/xmi.htm
[2] http://www.eclipse.org/modeling/gmf/
[3] http://www.eclipse.org/

TABLE II.
ICONS FOR CONNECTING TASKS WITH ACTIONS

Icons	Node
	Task
 <Start>	Start link
 <End>	Finish link
 <Solve>	Development link

Besides, the developer links each task with its characteristic actions (see Table II). These links can be of three types:

- Start link: An action begins the execution of a task.
- Finish link: The occurrence of an action involves the finishing of the task.
- Development link: An action helps to carry out a task but it does not start or finish the task.

In the final step, the repository structure is defined. The aim of this step is to design a repository to record actions and tasks. The repository records entries containing the action name, the task name and a set of descriptors. These descriptors are based on the guidelines that CSCW (Computer-Supported Cooperative Work) systems usually follow to notify changes in shared workspaces [19]. Some descriptors are used to give a complete definition of the characteristics of an action in a collaborative workspace:

- Time: It defines when the action is performed.
- Space: It shows the shared workspace where the action took place.
- Subject: It refers to the user who carried out the action.
- Object: It specifies the object in the shared workspace that was manipulated by the action.

Therefore, when a user carries out an action, the observation system inserts the abstraction of it through the descriptors into the repository.

C. *Specification of the solution model*

The previous procedures refer to the identification and processing of the actions and tasks representing the collaborative work (process) of the users of the CSCL environment. However, a comprehensive analysis also requires computational models of the solution built by the students for subsequent process-solution analysis.

In this context, it is necessary to define an application domain specifying all the elements and their relationships that can be manipulated by the students to build the solution. This procedure starts from the Dourish proposal [4] that defines an application domain by means of a set of central objects (called entities) interconnected through a set of links (called relationships), and a set of attributes that characterize the state of the entities and relationships of the domain.

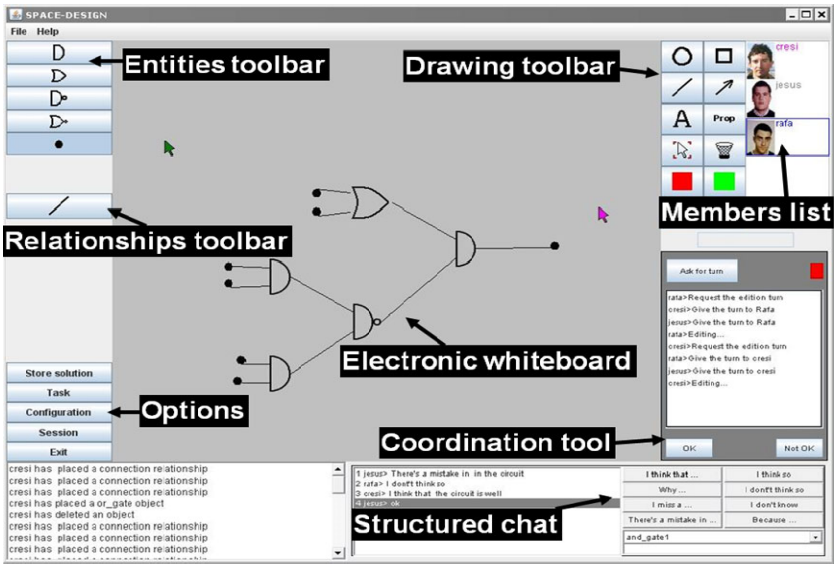

Figure 2. Modeling user interface of SPACE-DESIGN.

Once the developer of the observation system has identified the elements of the application domain, they are specified according to the MOF[4] meta-modeling standard. This specification provides an XML representation of the domain. Therefore, the observation system models the solution built by the users as a set of instances of elements from the meta-model. The solution model is updated when a new action interacts with the solution.

D. *Implementation*

In this procedure, the developer implements the observation mechanism. After defining the structure of the repository of actions/tasks and the solution model, the observation system is implemented on the basis of a three level architecture according to the following functionalities:

- Observation: The developer implements a set of functions to capture the actions that the user carries out through the user interface, which are characterized by descriptors. These functions should pay attention to interactions through user interface widgets, such us clicks, mouse movements, characters writing, etc.
- Classification: Its goal is to process each action, to categorize it according to the *diagram of actions*, to associate it to one of the defined tasks and to determine whether the action modifies the solution that is being built.
- Storage: This function updates the repository of information with new entries representing the actions affecting the collaboration process and/or the solution artifact.

IV. A CASE STUDY: SPACE-DESIGN

SPACE-DESIGN [5] is a distributed synchronous groupware system for modeling, i.e., the collaborative creation of diagrammatic models (see Fig. 2).

This system enables support to be provided to collaborative learning settings that follow a problem-solving approach. SPACE-DESIGN supports the design of diagrammatic models in different domains [7], which are defined by the user, but in this paper we focus on SPACE-DESIGN configured for solving problems in the domain of Digital Circuits. The main collaborative tools included in SPACE-DESIGN are (see Fig. 2):

- Electronic whiteboard: A shared surface where users create models (solutions). This includes toolbars with the entities and relationships of the application domain.
- Structured chat: This is used for communication between students. For this purpose, this includes a set of buttons with labels that express conversational acts (sentence openers). The chat can also be used as a conventional chat.
- Coordination tool: This is used by the students to coordinate the use of the whiteboard.

The approach proposed was applied in order to develop an observation system for integration into the SPACE-DESIGN environment. The following sub-sections describe how each one of the procedures proposed was applied.

A. *Identification of actions*

As a result of applying the first procedure, the first four levels of the hierarchical model are obtained (see Fig. 3):

- The root level refers to the SPACE-DESIGN environment.
- The second level identifies the three collaborative tools integrated into SPACE-DESIGN: modeling tool (electronic whiteboard), structured chat and coordination tool.
- The third level defines the types of actions (instrumental, cognitive, communicative and formal) that each tool supports according to the taxonomy presented (see sub-section III.A).
- The fourth level includes the specific actions within each action type, which can be identified with a specific name (e.g., insert object, modify object, proposal message, etc.).

[4] http://www.omg.org/mof/

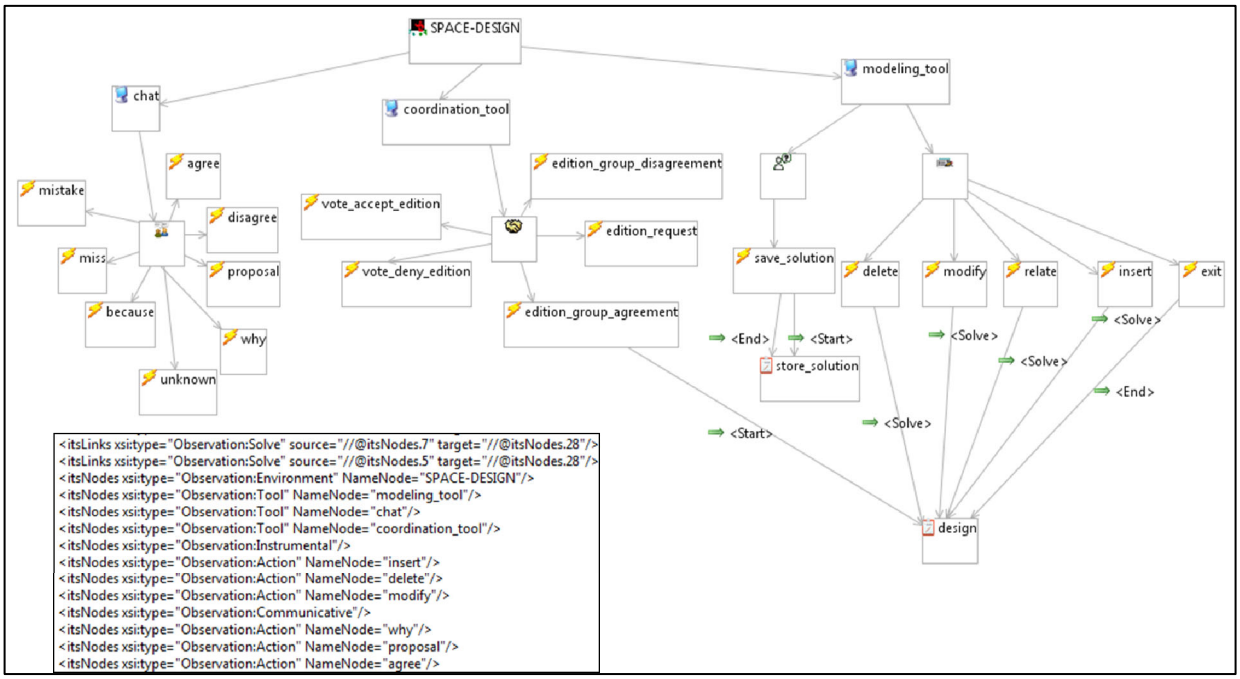

Figure 3. Excerpt of a *diagram of actions* with tasks designed and excerpt of XMI specification generated by the CASE tool.

B. *Repository*

The tasks that the students should carry out to solve the problem are identified here. The solution to the problem consists of two tasks that produce verifiable results: (i) the design of the digital circuit and (ii) the storing of the model designed as a solution to the problem.

In the second step of this procedure, the characteristic actions of each task are identified. When a user works on the collaborative whiteboard (first task), a set of modeling actions are produced. This first task finishes when a user saves the model (second task).

Fig. 3 shows an excerpt of the *diagram of actions* containing the actions and tasks. After making this design, the CASE tool automatically generates the XMI specification that represents the structure of the repository of actions and tasks (see Fig. 3).

C. *Solution model*

The solution model is specified through the identification of all operators involved in the application domain of Digital Circuits and the relationships that can be established between them. Then, the meta-model is specified. This defines the set of operators (*and*, *or*, *nand*, *nor* logical gates used in digital circuits) and the links (connections that propagate the output of a gate to the input of another one) that can be established between them.

The students use the CSCL environment to solve the problem through the building of a solution made up of instances of elements of the meta-model. The model, which contains the representation of the solution built, contains both semantic and graphic aspects. Besides, when the observation system processes a cognitive or instrumental action, it creates a new solution model in order to reconstruct the collaborative problem solving process.

D. *Implementation of the observation mechanisms*

The implementation of the observation mechanism is made up of three main modules. The *observer* is responsible for detecting the execution of the collaborative actions. For this purpose, the SPACE-DESIGN observer detects user interface events to identify actions according to the *diagram of actions* designed (see Fig. 3). The execution of these collaborative actions is communicated to the *classifier* using a set of descriptors (see sub-section III.B). For instance, the creation of a logical gate in the model is detected by means of the occurrence of a mouse click event. In this case, the descriptors that defined the space of these actions (see sub-section III.B) are the x/y coordinates of the logical gate into the electronic whiteboard and the object descriptors (see sub-section III.B) are the kinds of gates created.

The *classifier* processes the actions and, as a result, determines: (i) when the user is involved in the design of a digital circuit or in the storage of the model designed, and (ii) whether the actions insert, delete or modify a logical gate or a link in the digital circuit under construction.

Finally, the *storage* module updates the repository of actions and the solution model. The solution model is updated when an instrumental or cognitive action interacts with the solution.

V. CONCLUSIONS

This paper proposes a methodological approach to enable developers of collaboration analysis support systems in problem-solving CSCL environments to design observation mechanisms. The resulting observation mechanisms allow the processing of the interactions corresponding to the solution building process, linking actions with solution components and identifying the contribution of each user to the solution.

This approach has been tested through its application and implementation in the SPACE-DESIGN environment. In so doing, we have created diagrams that classify the

interactions supported by the environment and that enable the design of a repository containing the users interactions and of models that store solutions.

Currently, this approach is being validated with new case studies to verify its validity in other collaborative environments not aimed at the building of diagrammatic models. In these new case studies we will investigate further the automatic analysis of the solution built by the students.

In the future, the methodological approach will be completed with new procedures for the inference of analysis indicators to analyze the collaborative problem-solving work starting from the sources of information obtained in the observation; this corresponds to the *abstraction* analysis phase.

REFERENCES

[1] N. Avouris, V. Komis, M. Margaritis and G. Fiotakis, "An environment for studying collaborative learning activities," *Educational Technology & Society*, vol. 7 (2), pp. 34-41, 2004.

[2] C. Bravo, M.A. Redondo, M.F. Verdejo and M. Ortega, "Framework for Process and Solution Analysis in Synchronous Collaborative Learning Environments," *International Journal of Human-Computer Studies*, vol. 66 (11), pp. 812-832, 2008. (doi:10.1016/j.ijhcs.2008.08.003)

[3] A. Dimitrakopoulou et al., "State of the Art on Interaction Analysis: Interaction Analysis Indicators". Kaleidoscope Network of Excelence. Interaction & Collaboration Analysis Supporting Teachers and Students' Self-Regulation. Jointly Executed Integrated Research Project. Deliverable D.26.1, 2004.

[4] P. Dourish, "Using meta-level techniques in a flexible toolkit for CSCW applications," *ACM Transactions on Computer-Human Interaction*, vol. 5, 2, pp. 109-155, 1998. (doi:10.1145/287675.287676)

[5] R. Duque, J. Gallardo, and C. Bravo, "Defining tasks, domains and conversational acts in CSCW systems: the SPACE-DESIGN case," *Journal of Universal Computer Science*, vol. 14(9) pp. 1463-1479, 2008.

[6] W. Gaaloul, S. Alaoui, B. Baina, K. and C. Godart, "Mining Workflow Patterns through Event-Data Analysis," *Proceedings of the 2005 Symposium on Applications and the internet Workshops*, pp. 226-229, 2005.

[7] J. Gallardo, C. Bravo and M.A. Redondo, "Developing Collaborative Modeling Systems Following a Model-Driven Engineering Approach," R. Meersman, Z.Tari and P. Herrero (Eds.): *Proceedings of the Workshop on System/Software Architectures at OTM Confederated International Workshops*, Monterrey, Mexico, pp. 442-451, 2008.

[8] A. Inaba, T. Tamura, R. Ohkubo, M. Ikeda, R. Mizoguchi, R. and J. Toyoda, "Design and Analysis of Learners' Interaction based on Collaborative Learning Ontology," *Proceedings of the 2nd European Conference on CSCL*, pp. 308-315, 2001.

[9] T. Koschmann, A. Kelson, P. Feltovich, and H. Barrows, "Computer-supported problem-based learning," CSCL: Theory and Practice of an Emerging Paradigm T. Koschmann (Ed), pp. 83-124, 1996.

[10] Y. Li, J. Wang, J. Liao, D. Zhao, and R. Huang, "Assessing Collaborative Process in CSCL with an Intelligent Content Analysis Toolkit," *Seventh IEEE International Conference on Advanced Learning Technologies*, pp. 257-261, 2007.

[11] W. Lowe, "Software for Content Analysis – A Review," *http://people.iq.harvard.edu/~wlowe/Publications/rev.pdf* , 2002. [Accessed 09.01.2009].

[12] A. Martínez, Y. Dimitriadis, and P. de la Fuente, "Interaction analysis for formative evaluation in CSCL," *Computer and Education: Toward a Lifelong Learning Society*, Kluwer, The Netherlands, pp. 227-238, 2003.

[13] A. Martínez, Y. Dimitriadis, and P. de la Fuente, "Towards an XML-based Representation of Collaborative Interaction," *Proceedings of the Computer Support for Collaborative Learning, CSCL*, Bergen, pp. 379-384, 2003.

[14] L.D. Miller, A. Eck, L.K. Soh, and H. Jiang, "Statistics and analysis tools for a computer-supported collaborative learning system," *Proceedings of Frontiers in education conference - global engineering: knowledge without borders, opportunities without passports*, pp. 1-6, 2007.

[15] M. Mühlenbrock and U. Hoppe, "Computer supported interaction analysis of group problem solving," *Proceedings of CSCL'99*, pp. 398-405, 1999.

[16] M. Mühlenbrock, "Automatic action analysis in an interactive learning environment," *Proceedings of the workshop on Usage Analysis in Learning Systems at the 12th International Conference on Artificial Intelligence in Education AIED-2005*, Amsterdam, The Netherlands, pp. 73-80, 2005.

[17] J.W. Sarmiento and G. Stahl, "Bridging and Persistence in Sustained, Collaborative Problem Solving Online," *Proceedings of the 40th Annual Hawaii international Conference on System Sciences*, pp. 78, 2007.

[18] A. Soller, A. Martínez-Monés, P. Jermann and M. Muehlenbrock, "From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning," *International Journal of Artificial Intelligence in Education*, vol. 15 (4), pp. 261-290, 2005.

[19] J. Tam. and S. Greenberg, "A Framework for Asynchronous Change Awareness," *International Journal of Human Computer Studies*, vol. 64 (7), pp. 583-598, 2006. (doi:10.1016/j.ijhcs.2006.02.004)

AUTHORS

R. Duque is a researcher in the Department of Information Technologies and Systems, University of Castilla-La Mancha, Ciudad Real, 13071 Spain (e-mail: rafael.duque@uclm.es).

C. Bravo belongs to the Department of Information Technologies and Systems, University of Castilla-La Mancha, Ciudad Real, 13071 Spain (e-mail: crescencio.bravo@uclm.es).

J. Gallardo is with the Department of Information Technologies and Systems, University of Castilla-La Mancha, Ciudad Real, 13071 Spain (e-mail: jesus.gallardo@uclm.es).

W. J. Giraldo is with the Systems and Computation Engineering, University of Quindío, Armenia, Colombia, (e-mail: wjgiraldo@uniquindio.edu.co).

M. Ortega is a Professor (Computer Systems and Languages) and head of the CHICO research group (Department of Information Technologies and Systems) at the University of Castilla-La Mancha, Ciudad Real, 13071 Spain (e-mail: manuel.ortega@uclm.es).

This research is supported by the Comunidad Autónoma de Castilla-La Mancha (Spain) in the PCI08-0069-7887 and PAC07-0020-5702 projects, and by the Ministerio de Educación y Ciencia (Spain) in the TIN2005-08945-C06-04 project.

This article was modified from a presentation at X International Symposium on Computers in Education (SIIE2008) 1st-3rd October 2008, Salamanca, Spain. Manuscript received 12 January 2009. Published as submitted by the authors.